Web Watch: A Method for Monitoring Changes on the Web

~~Web Monitoring Proposal~~

Alexander Tuzhilin and Ravi Arunkundram

1. Introduction

The Web is a very dynamic and constantly evolving phenomenon: new Web pages are constantly created, and old pages are frequently modified. Therefore, it is often very difficult to follow all the changes made at different Web sites, especially if you want to keep track of the changes made at numerous sites. Even if one can follow simple changes, such as addition and deletion of certain keywords at a site, it is much more difficult to follow complicated changes and relationships across multiple sites. For example, it may be difficult to detect the situation when several companies suddenly created links to each other's Web site following an introduction of a new product by one of them. Nevertheless, it may be important to be able to detect such changes because some of them contain important information about new developments in or among the organization (for example, once we detected that the companies started pointing at each other's Web sites, it may be an indication that they are forming an alliance, and itmay be useful to know that).

To deal with the problem of monitoring changes at Web sites, some vendors developed systems that provide simple monitoring capabilities. For example, AltaVista, Excite

Unfortunately, Lycos monitoring capabilities are quite simplistic and do not allow monitoring more complex relationships, such as the one described above.

This invention presents a method for monitoring complex changes occurring at one or several the Web sites. We believe that it will be useful to a broad range of Web users, such as financial analysts, industry analysts, investors, regulatory agencies, and advertisers.

In the next section we describe a monitoring and probing specification language, and in Section 3 we explain how this language can be implemented. Finally, in Section 4 we will describe its practical embodiments.

2. Monitoring/Probing Specification Language

The general structure of a monitoring/probing rule (or a trigger – we use these terms interchangeably in this document) is

WHEN <events>
IF <conditions>

THEN <probing-actions>

where events, conditions, and probing-actions are briefly defined as follows (they will be described in detail later).

Event: An event is a change in state of one or more objects that we are monitoring. For instance, an appearance or disappearance of one or more keywords, the appearance of a new link between two sites, a change in physical attributes of a web page are examples of events.

Conditions: Conditions are constraints that apply to events and act as filters that refine the space of events to result in a smaller subset that are of interest to entities that monitor the web. For instance, the appearance of the keyword "push technology" is an event that is of interest only if the keyword "cable modem" does not exist anywhere on the site (the qualifier "only if cable modem does not exist anywhere else" is a condition that is applied to the event).

Probing Action: The purpose of the probing action is to investigate what is "going on" at the site or a collection of sites once the events in the WHEN-clause occurred and satisfied the conditions in the IF-clause. For instance, the system may explore what is "going on" at a predefined list of web sites when the keyword "cable modem" recently appeared in one or more of them (if the events and condition specified above hold good). A probing action usually returns some information that is the result of the examination of one or more web sites in order to understand deeper reasons for the occurrence of events in the WHEN-clause and the conditions in the IF-clause.

The following examples illustrate the structure of WHEN-IF-THEN rules:

1. WHEN the Keyword "Cable Modem" Appears on a site
   IF the Keyword "Push Technology" Does not appear at the Site
   THEN Find out on how many sites (from a pre-defined list of sites) the word "Cable Modem Appeared" and of these how many feature the keyword "Push technology"

2. IF (Several Web sites are linked into a Star[1] configuration AND the Keyword "Cable Modem" is found on all the nodes in the configuration)
   THEN (Find all such configurations, the members of which contain the word "Cable Modem").

[1] "Star" is a user-defined predicate, intuitively, meaning that there is a central site that is linked to all other sites.

3. IF (A keyword set consisting of {"Push technologies", "Web TV", "Channels"} appears in all the nodes (members) of a user-defined configuration)
   THEN (Find all such configurations where such keywords appeared; return the list of the corresponding URL's).

4. IF (the number of visits to a site increase at a rate greater than *r* AND the sites contain the keyword set {Cable Modems, Web TV})
   THEN (find all sites that contain the same keywords and whose visit rate was greater than r; return the set of corresponding URL's).

5. IF (a keyword appears at a site AND links to that site from other sites disappear)
   THEN (find all the sites from which links to this site disappeared;
   find if these sites have any keywords in common;
   find all the sites which are now linked to this site;
   find if these sites have any keywords in common;
   return four lists)

The WHEN- and IF-clauses form the *monitoring part* of the rule (trigger), and the THEN-clause forms the *probing* part of the rule. The monitoring part of the trigger watches for certain events to occur that satisfy certain conditions. Once the monitor detects appropriate events, it does not necessarily informs the user about this for the reasons explained below. It tries to explore the situation in which these events occurred by executing certain probing actions specified in the probing part of the trigger (i.e. in the THEN clause). These probing actions will be described below. The structure of a rule is similar to the structure of triggers used in active databases. However, the nature of events, conditions and especially probing actions is substantially different, as will be explained below.

An event in the WHEN-clause may be ***atomic*** or ***composite***. A composite event is the combination of two or more atomic events. One of the most popular methods to define composite events is through the conjunction of atomic events, i.e. as E1 and E2 and ..., and En, where E1, E2, ..., En are atomic events, such as "link between two sites appears," "a keyword appears within a site," or "a page (or link) was deleted." However, composite events can be defined in more general terms as well (e.g. as a Conjunctive or a Disjunctive Normal Form (CNF or DNF) of its atomic events). Atomic events belong to one of the following classes (but atomic events are not limited just to these classes):

1. A link appears/disappears
2. A Keyword (or a group of Keywords) appears on a page.
3. A Page's text is modified.
4. Change in the physical attributes of a page.
5. A visitor visit a page.

Although events in the monitoring part of the rule are useful, it is not necessary to use them in general for monitoring Web activities. The reason for that is that they can be

simulated through the conditions of the IF-clause. In our model, (as used in Web Watch - - reference???), we propose to model events by specifying certain conditions in the IF clause using the methods known to a person having ordinary skills in the art. This approach is adopted for expositional simplicity and for use in the creation of a prototype and it is essential to note that this is only a *particular instance of a general rule* that we propose. Therefore, we will focus only on the structure of conditions and probing-actions in the rest of this section.

Conditions:

One way to specify conditions in the IF-clause of a rule would be through a conjunction of atomic conditions, i.e. as P1 and P2 and .... and Pn, where Pi is an atomic condition. An example of an IF-clause is

IF Site S_1 is Linked to Site S_2 AND keyword K_1 appears in S_2 AND Keyword exists in S_1

where "Site S_1 is Linked to Site S_2," "keyword K_1 appears in S_2," and "Keyword exists in S_1" are atomic conditions.

In order to define atomic conditions, it is necessary to explore the modeling primitives (or the basic entities) used in our system. These are:

1. **Web Pages**: A web page is a collection text, pictures, video and audio objects (and other media) that are on the Web and share the same URL. Two different Web pages will not have the same URL. A Web page may have physical attributes such as size, date of last change, title etc. We will refer to Web pages as 'Pages' in the rest of the document.
2. **Web Sites**: A Web Site is a collection of one or more Web pages. Pages within a site are linked by links. A Web Site also has a URL which uniquely identifies the site. We will refer to these as Sites in the rest of the document.
3. **Key Words**: These are text objects (words, phrases etc.) found in a web page and therefore, by extension found in a Web site.
4. **Links**: Links are embedded in Web pages and allow a visitor to move from one location to another. A Link may take a user from one page to another or may simply take her to another location on the same page.
5. **Visit**: Refers to the act of a visitor (anybody who reaches a Web Site) visiting a site.

In addition to these basic entities, we also use some of the composite entities that can be derived from the basic entities using some data definition language (for example, if the basic entities are defined in terms of a relational data model as tables, the composite entities are defined as relational views, i.e., as SQL expressions). Examples of these composite entities are:

1. **Configuration**: A collection of Web Sites that are linked to each other. A

Configuration can have one (trivial case) or more Web Sites. For example, a set of sites linked in a circular fashion forms a "ring" configuration; or a configuration consisting of a central site that is linked to all other sites in the configuration and other sites being linked only to the central site, forms a "star" configuration.

2. **Reach relation**: It specifies which sites can be reached from a given site. This relation can be defined as a transitive closure of the LINKS relation.

Basic and derived entities are modeled as data objects in the corresponding data model. For example, if the data model is relational, then these entities are modeled as relations (predicates). If the data model is object-oriented, then they are modeled as data objects. Without loss of generality, we assume that the model is relational data model and that we model these entities as predicates (using relational terminology, we will sometimes call them "relations" or "tables" and assume that these terms are synonyms). For example, LINKS is a predicate of the form LINKS(site1,site2), where site1 is the site from which the link originates and points to site2. Moreover, some of the predicates are temporal because they (implicitly) change over time. For example, predicate LINKS may implicitly change over time (because links can be added and removed over time). Note that this invention is not limited to the relational data model and is applicable to other data models as well.

Once we defined basic entities of the data model (e.g. as predicates/relations), we specify the atomic conditions that can appear in the IF-clause of the monitor as follows. An atomic condition can either be a predicate, or its negation, or a past temporal predicate (a past temporal predicate is either a temporal predicate preceded by a unary past temporal operator or a binary temporal operator). Some examples of unary past temporal operators are Always_in_the_past, Sometimes_in_the_past, Sometimes_within_the_past(T_time_units), Always_within_the_past(T_time_units), and an example of a binary temporal operator is Since. Some examples of atomic conditions are:

1. Key_Words(site,keyword,count) meaning that the keyword "keyword" has count "count" at site "site".

2. "LINKS(site1,site2) Sometimes_within_the_past(2weeks)" -- meaning that a link existed at some point between sites site1 and site2 within the past 2 weeks. Note that LINKS is a temporal predicate because it implicitly changes over time.

3. "Key_Words(XYZ, "cable_modem", count) Since LINKS(XYZ, ABC)" -- meaning that the keyword "cable modem" appeared at cite XYZ sine the link was established between sites XYZ and ABC.

Probing Actions

When the monitoring condition of a trigger is satisfied, the user can be notified about this event and the monitoring results can be returned to him/her. Unfortunately, these

notifications can overwhelm the user when he/she wants to monitor many sites. For example, if the user wants to monitor 500 sites, and each site sends the user 3-4 notifications a day, this means that the user will get 1500 – 2000 alarms a day (which can be quite alarming to him/her indeed!).

To address this problem, it is important to explore further what is "going on" when the monitor of a trigger detects the appropriate change at the site(s), and this is the purpose of the probing part of the trigger (that is specified in the WHEN-clause of a rule).

The THEN clause consists of a sequence of probing operations. We consider the following types of probing operations (although this invention does not depend only on these operations, and other operations can be considered as well):

1. Issue an alarm and return appropriate information detected by the monitoring component of the trigger back to the user. For example, the trigger

   IF the keyword "Cable Modem" appears at a site but the keyword "Push Technology" does not appear at that site
   THEN issue an alarm and return that information to the user

   returns the names of all the sites where the keyword "Cable Modem" appears but not the keyword "Push Technology."

2. Retrieve information either from the Web site(s) or from the temporal predicates maintained at the monitoring site in a way to be described in Section 2.1 and check if it satisfies certain conditions. One way to do this is by formulating SQL or other types of temporal or non-temporal queries. For example, assume that in the monitoring part of a trigger it was detected that a certain configuration among a group of sites was broken when a certain keyword appeared in one or more of these sites. Then in the probing part we may want to find the list of all sites whose links to those sites where the keyword appeared were deleted after the appearance of the keyword. This probing action can be expressed with the following three SQL queries:

(The first query extracts the list of all sites that point to one particular site (e.g., www.target.com) in the previous monitoring period (before the keyword appeared))

```
SELECT [Web Links Data].[Origin of Link], [Web Links Data].[Target Site],
    [Web Links Data].[Monitoring Period], [Web Links Data].Status
FROM [Web Links Data]
WHERE ((([Web Links Data].[Target Site])="www.target.com") AND (([Web
    Links Data].[Monitoring Period])="T - 1") AND (([Web Links Data].Status) =
    Yes));
```

(The second query extracts all those sites that do not have a link with that site in the present monitoring period and stores it as a temporary table called "TEMP: Sites with no current link")

```
SELECT [Web Links Data].[Origin of Link], [Web Links Data].[Target Site],
  [Web Links Data].[Monitoring Period], [Web Links Data].Status
FROM [Web Links Data]
WHERE ((([Web Links Data].[Target Site])="www.target.com") AND (([Web
  Links Data].[Monitoring Period])="T") AND (([Web Links Data].Status)=No));
```

(The third query joins the two temporary tables in an SQL Query and extracts the list of those sites that have deleted a link in the current monitoring period)

```
SELECT [TEMP: Sites with no current link].[Origin of Link]
FROM [TEMP: Sites with no current link] INNER JOIN [TEMP:sites with earlier
  link] ON [TEMP: Sites with no current link].[Origin of Link] = [TEMP:sites
  with earlier link].[Origin of Link]
GROUP BY [TEMP: Sites with no current link].[Origin of Link]
ORDER BY [TEMP: Sites with no current link].[Origin of Link];
```

3. Execute a program written in a general-purpose programming language (e.g. C) that examines certain conditions. For example, such program may visit different Web sites in some sequence and return the URLs of those sites that satisfy certain properties (e.g. have increasingly higher visitation traffic).

4. Execute a Data Mining Query ([IVA96], [H+96],[SOMZ96],[K+94]). A data mining query discovers all the patterns of a certain type (specified as a set of constraints on the types of patterns to be discovered). It is stated in a Data Mining Query Language (e.g. M-SQL [IVA96], DMQL [H+96], as a meta-query [SOMZ96], or as a template of [K+94]) and returns a set of patterns satisfying this query. For example, a data mining query may want to find all the patterns (e.g. expressed as association rules [AIS93]) that correlate appearances and disappearances of the keywords "Cable Modem" and "Push Technologies" across certain specified groups of sites. The discovered patterns are returned to the user. The returned patterns may provide a better insight into the relationship between these terms and, more generally, better insights into what is "going on" with these two technologies.

To illustrate how the composition of these basic operations works, consider Example 5 from Section 2. The THEN-clause of the trigger from Example 5 consists of a sequence of 5 operations (that can be defined with operations of type 2, i.e., retrieve information operations that can be defined, for example, with SQL queries).

3. Implementation Issues

An important issue is how to implement the Web monitoring triggers described in Section 2.

One of the main implementation issues is that Web is *stateless* in the sense that it does not "remember" its history. For example, there is no way to test whether the volume of

visits at the site has been monotonically increasing within the past 3 months unless the user maintains the information about the volume of visits over time him/herself (the Web site does not maintain this information itself). Therefore, it is the responsibility of the user to maintain the necessary information. Of course, the main question is what historical information needs to be maintained and where for the monitoring purposes.

3.1 Information to be Maintained at the Monitoring Site

Given a set of triggers, we need to maintain the (temporal) predicates that would allow us to evaluate temporal expressions contained in these triggers . This means that we need to store the current values of these predicates and, in certain cases, their past histories. For example, assume that we have temporal predicate "LINKS(site1,site2) within_past(2months)" specifying that a link existed between site1 and site2 at some time within the past 2 months. To be able to monitor and evaluate such a predicate, we need to maintain the current state of predicate LINKS (at the present time) as well as its past history within the last two months. Since time is continuous, and we can evaluate the presence of links between site1 and site2 only at some finite set of times, we need to specify the sampling (monitoring) rate for the evaluation of predicate LINKS and check the presence of links at the time points corresponding to this sampling rate. For example, we may decide to check the status of predicate LINKS every day, or every 8 hours, or every hour. Then we assume that no abnormal changes happen to predicate LINKS between the sampling points. For example, assume that we sample predicate LINKS every day. Then if the link between site1 and site2 existed yesterday and still exists today, this means that nobody deleted *and* inserted the same link within the last day (but it is of course possible to delete the link during that day; in this case, the monitor detects the change at the next sampling point). Similarly, if the temporal predicate is "always_past LINKS(site1, site2)," stating that there was always a link between site1 and site2 in the past, then we may want to maintain two predicates LINKS(site1, site2) and ALWAYS_PAST_LINKS(site1, site2) in order to monitor condition "always_past LINKS(site1, site2)".

This additional information about the past histories of temporal predicates appearing in user-defined triggers can be stored either locally at the user's site (in a distributed way) or it can be stored centrally at the same site for various users (e.g. at the site of the company's that sells this Web monitoring software). This invention is not restricted to a specific storage method and encompasses both methods. Moreover, when we want to mention the site where these past histories of predicates are stored, we will refer to that site as a "*monitoring site*", assuming that it could be either the user's site or the central site used by all the users.

In general information to be maintained at the monitoring site can be determined as follows. Take the set of all the temporal predicates appearing in the monitoring parts of all the triggers. If the predicate is non-temporal (no trigger refers to its past history), we just store the current copy of that predicate at the monitoring site (the copy at the latest sampling period). If the predicate is temporal (some triggers refer to its past history), we store the current copy of that predicate and the following additional information. For each

temporal operator (e.g. Always_past, Sometimes_past, etc.) we store an *auxiliary* copy of the predicate pertaining to this operator. For example, for the temporal expression Always_past LINKS(site1, site2) appearing in one of the triggers, we store the current copy of LINKS(site1, site2) and an auxiliary predicate Always_Past_LINKS(site1, site2). Always_Past_LINKS(site1, site2) is true if and only if there was always a link from site1 to site2 in the past. Similarly, we store an auxiliary predicate Sometimes_Past_LINKS for the temporal expression "Sometimes_Past LINKS". In general, we maintain one auxiliary operator for *every* (predicate, temporal operator) pair appearing in one of the monitoring triggers created by the user. These auxiliary predicates can be maintained using the methods known to the person having an ordinary skill in the art (e.g. as described in [Chom92]). For example, predicate Always_Past_LINKS is updated every monitoring period by removing records (site1, site2) from it that do not appear in the current copy of LINKS(site1, site2). Note that these auxiliary predicates are *non-temporal* (they only simulate temporal predicates but do not have any temporal component as part of their structure) and can be stored using the conventional methods known to someone with an ordinary skill in the art.

One of the steps in predicate maintenance is the process of obtaining the current copy of the predicate. This means visiting the monitored site(s) and evaluating the present conditions of the predicate(s) at that site. One of the central issues in this process is what information should be checked and, even more importantly, what information should be brought from the monitored site(s) back to the monitoring site. This is a critical issue because, we want to monitor many sites and need to minimize the data to be transferred back to the monitoring site in order not to clog the communication lines. We will address this issue in the next section.

3.2 Efficient Methods of Transferring Data to the Monitoring Site

As was demonstrated in Section 3.1, for the monitoring purposes, it is necessary for the system to bring back at each monitoring period only the current states of the predicates that are maintained at the monitoring site, such as LINKS, Web_Sites, Key_Words, etc. Given this data, the past temporal predicates can be evaluated using the methods described in Section 3.1 (e.g. maintaining LINKS and Always_Past_Links predicates).

To obtain the current copies of these predicates, the agents should be created that visit the sites being monitored and retrieve the corresponding predicates. For example, if we monitor LINKS(site1,site2), then an agent should go to site site1 and determine if there is a link from that site to site2. Such agents are created at the time of system development using the methods known to a person having the ordinary skill in the art and are *not* written by the user at the time monitors are specified by him/her.

An additional optimization issue arises when one can bring *less* data to the monitoring site than in the current copy of the predicate being monitored. For example, assume that the temporal predicate being monitored in the IF-clause of a trigger is "Always_Past LINKS(site1, site2)". As was explained in Section 3.1, we can monitor this predicate by creating an auxiliary predicate Always_Past_LINKS(site1,site2). Note that this predicate

is monotonically decreasing over time. Also note that rather than retrieving the relation LINKS in its entirety at each monitoring period, it is sufficient to retrieve only the records from the auxiliary predicate Always_Past_LINKS (wich is a subset of LINKS).

In general, if auxiliary predicates are monotonically decreasing (such as Always_past_LINKS), then we retrieve these predicates rather than the original predicates (e.g. LINKS). To the contrary, if auxiliary predicates do not decrease over time (such as Sometimes_Past_LINKS), the *original* predicates are retrieved from the Web sites being monitored.

~~5.2.5.~~ How to Build the Probing Component

QUESTION: DO WE NEED THIS SECTION

As was stated in Section 2, the probing component supports four types of probing operations: user notification, information retrieval, program execution, and data mining querying operations. User notification and program execution operations can easily be implemented by a person having ordinary skills in the art. Therefore, we will focus on the information retrieval and data mining querying operations in the rest of this section.

Information retrieval. We need to distinguish between the operations retrieving data from the Web and those retrieving the historical data from the monitoring site. The former can use any of the existing Web-related information retrieving methods known to a person with ordinary skills in the art. The latter can use any database information retrieval methods, such as the utilization of any of the existing database query languages. For example, historical data can be retrieved from the monitoring site using SQL as illustrated in Section 2. The implementation issues related to retrieving historical data are well-known and can be implemented by a person having an ordinary skill in the art.

Data mining query languages. There are several existing data mining query langages, as discussed in ~~Section ???~~ and they can be used for the implementation of this probing operator. the "probing actions" section

4. Practical Embodiments

We believe that there are two reasons for monitoring a Web site. The first is the obvious direct benefit to be derived from acquiring the information that may be contained in one or more web sites. To have access to large and distributed information repositories on the web, it is necessary to know what information resides at a site. Most commercially available search engines, like AltaVista and Yahoo provide this service. There is a second reason for monitoring the Web which has to do with the significance that users may attach to events that occur on the Web or *Web Events*. A Web Event is *some change* in

state of a web site (or sites) that may have causal implications to users. This when re-stated means that users may find the actual contents of the sites and the navigational patterns associated with them (links to other sites) less significant than the change in contents or the appearance or disappearance of links. We refer to information about such changes as ***meta information.*** To understand how meta information is of very great value to variety of business entities, Investors, Venture Capitalists, Consulting firms, economic and financial research agencies, investment bankers and advertisers to name some, consider the following example. Let us a consider a microprocessor making firm (say, Alpha Corp.) that is in the race for putting more transistors on a chip by using a process called Photolithography. This firm is a part of an alliance that has decided that Photolithography and associated technologies will be the commercially successful route to take in the future. This firm's web site is connected (linked) to the web site of the firms that are its alliance partners in this business. A competing technology using high energy X-rays is used by another consortium of companies (say Beta Corp.) whose web sites are also linked. Analysts and investors in the microprocessor industry, watch developments closely in the field and wish to be informed of the alliances between companies that offer different technologies. If a new firm or a group of firms start offering a technology based on photolithography that complements Alpha Corp.'s products, this information is of considerable significance to investors and analysts. They could use our system, WebWatch to monitor the appearance of Keywords "photolithography" on the Web. Further, they could define the triggers in such a way that WebWatch could identify the entry of these new firms into the field and probe further to see if they are connected to Alpha Corp.'s alliance or if these firms have created a competing alliance (by forming an inter-linked configuration of sites). Suppose, links to one or more of the firms from Alpha corp.'s web configuration were to disappear from the web sites of other firms in the configuration, WebWatch could immediately recognize this as an event and probe for more details to determine possible causes for the removal of these links. The system's probe could identify the following causes:

(i) The firms offer a competing technology or are aligned with a competitor: WebWatch could look for keywords that indicate that these firms are now offering competing products or it could look for links from the sites of Beta

Corp.'s alliance to the sites of these firms to determine if they have joined a competing alliance.

(ii) These firms have formed a alliance of their own and are now in direct competition with Alpha Corp.; WebWatch would look for configuration formation consisting of thee firms (and perhaps a few others) that resulted in the links from Alpha's alliance being lost.

Once the system's probe finds a pattern it could immediately alert the user and describe its findings through electronic mail. The kind of information that can be discovered by the system's probe would include, entry of new firms or consortiums of firms, formation of alliances, new products or services being offered, sites that have attracted many users, ideas products and services that attract consumer attention etc. A regulatory agency could use this information to see if there is need to investigate antitrust issues in formation of anticompetitive alliances. An advertiser could monitor some sites to see if there is a sudden spurt in the hits to that site after a keyword has been introduced on the site. He could also find out if the spurt in visits to the site happened after the addition of links from a particular site or if it is correlated to with some events. This information would help the advertiser asses the worth of an advertisement on the site. Analysts and venture capitalists who are monitoring the microprocessor industry and watching for four key leading technologies, (say, High energy X-rays, Photolithography, extreme ultraviolet and Metal Pairing) could effectively monitor the entry of new firms, the positioning of competing products, the complimentary or competing services offered by firms, the number of visits that these firms and their alliances generate etc. Economic and financial research agencies could use the WebWatch to determine changes in pricing and promotion policies (as indicated by keywords on sites), the attention generated by new developments (hits to a site) macro trends in the industry (more firms offer High energy X rays rather than Metal Pairing) and changes in strategies.

The *coupling of monitoring with probing* offers two important advantages. The first is prevention of information overload. WebWatch does not immediately alert the user as soon as a minor change in the domain if his interest takes place. The system continues to probe to see if this change is accompanied by events of significance (pre-defined by the user) and returns a the list of those changes. Thus, it avoids a huge information overload

that makes most simple monitoring mechanisms less than useful. The second advantage is that it allows the user to define a relationship between certain events and their possible implications. The occurrence of an event (deletion of a link to a site) may be significantly correlated with one phenomenon (addition of a link from different site) but may be entirely meaningless in the context of other phenomena (say, spurt in the number of visits to the site). WebWatch allows the user to define triggers that make an explicit connection between what is symptomatic event and what needs be further investigated.

References

[AIS93] Agrawal, R, Imielinsky, T. and Swami, A. 1993 .....

[Chom92] Chomicki, J. History-Less Checking of Dynamic Integrity Constraints., 8th IEEE Conference on Data Engineering, 1992.

[H+96] Han, J. et al. .....

[IVA96] Imielinsky, T., Virmani, A., and Abdulghani, A. 1996. .....

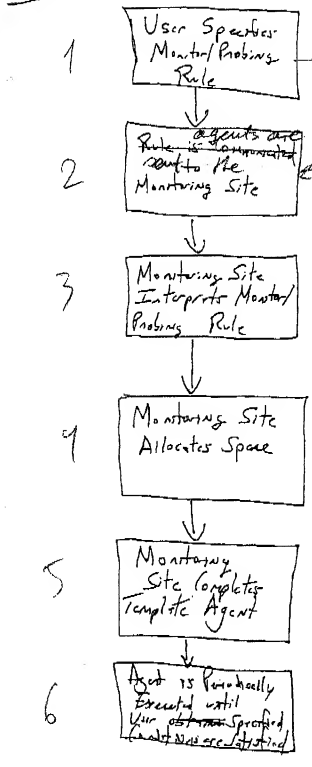
[K+94] Klemettinen, M., Mannila, H. et al. .....

[SOMZ96] Shen, W.-M., Ong, K.-L., Mitbander, B, and Zaniolo, C. .....

[T+93] Tansel, A. et. al. Temporal Databases, Benjamin/Cummings, 1993.

********** TO DO LIST **********

Steps


1
User Specifies Monitor/Probing Rule
2
~~Rule is communicated~~ ~~agents are~~ sent to the Monitoring Site
3
Monitoring Site Interprets Monitor/Probing Rule
4
Monitoring Site Allocates Space
5
Monitoring Site Completes Template Agent
6
Agent is Periodically Executed until User ~~obtains~~ Specified ~~Result~~ Data are satisfied
Rule is parsed (proposed) and are of general monitoring agents, general ~~Also storage~~ facilities at client station are allocated.

DRAFT

General description section

1. Overview of the main idea
   the idea of monitoring changes to Web sites;
   overview
   what will be described in the rest of the document.

2. Monitoring specification language
   a. monitoring component -- the IF clause (say that we don't want to monitor events or activities -- no WHEN and WHILE clauses; explain why we decided not to do this).
   b. probing component -- 4 parts: data mining queries, SQL queries, event state changes, return discovered info (or notification) back to the user

3. Implementation
   a. Maintaining history -- explain why important (statelessness of Internet); Problem: given the set of monitors, what needs to be stored in histories, how to store this information (what are the data structures) and how often this information needs to be brought back
   b. Optimization issues: how to bring back a *minimal* amount of data that still satisfies our monitoring needs. Also, *what* needs to be brought back to our site.
   c. How this information can be brought back to the site. Launch an agent to do this. With what information this agent should be provided before it "goes" to the site? How this information should be supplied to the agent? What are the algorithms?
   d. How to implement the probing component? What are the mechanisms to implement the probing?